# Scheduling and Parallelism in RF Benchmarks: Throughput Without Drift

Benjamin J. Gilbert
Spectrcyde RF Quantum SCYTHE, College of the Mainland
Texas City, TX 77590
Email: bgilbert2@com.edu
ORCID: 0009-0006-2298-6538

***Abstract*—Benchmarking RF demodulation pipelines is often bottlenecked by slow simulation loops. While multi-threaded execution can reduce wall-clock time, naive scheduling may bias robustness estimates if seeds are not managed consistently. We demonstrate a scalable scheduling harness that exploits CPU worker parallelism and batch sizing to accelerate benchmarks without introducing statistical drift. Using the `SignalIntelligenceSystem` threading architecture, our results show near-linear throughput scaling (up to 8× speedup) while preserving repeatability across seeds. This enables practical deployment of agentic sweeps and ghost mode analysis within operational time budgets.**

## I. Introduction

RF benchmarking campaigns involve hundreds of simulation runs across SNR, $\Delta f$, modulation depth, and other parameters. Each run can take seconds to minutes, making exhaustive parameter sweeps prohibitively expensive [1]. While probabilistic agentic sweeps reduce the number of required evaluations, and ghost mode analysis quantifies the economic cost of false positives [2], both approaches still require substantial computational resources for practical deployment.

To make agentic sweeps and ghost analysis feasible within operational budgets, we must reduce wall-clock runtime without biasing results. Naive parallelization can introduce statistical drift if random seeds, timing dependencies, or shared resources are not managed carefully. This paper studies how CPU worker scaling and batch sizing affect both throughput and reproducibility in RF benchmarking workflows.

Our approach leverages the existing threading architecture in the `SignalIntelligenceSystem` from `core.py`, extending its `_signal_processing_loop` and `_data_collection_loop` pattern to support batch-parallel execution across multiple worker threads. Results demonstrate near-linear scaling up to 8 workers with zero statistical drift when proper seeding protocols are followed.

## II. Methods

### A. SignalIntelligenceSystem Threading Architecture

The `SignalIntelligenceSystem` in `core.py` already implements separation of concerns through independent signal processing and data collection threads. This design provides a natural foundation for parallel batch execution. We extend this architecture by spawning $W$ worker threads, each consuming batches of parameter configurations from a shared queue.

```python
import concurrent.futures
import numpy as np
from SignalIntelligence.core import
    SignalIntelligenceSystem

def run_parallel_benchmark(configs,
    batch_size=16, workers=8):
    sis = SignalIntelligenceSystem(config,
        comm_network)

    def process_batch(batch_with_seed):
        batch, seed_offset = batch_with_seed
        np.random.seed(seed_offset)  #
            Deterministic seeding
        results = []
        for cfg in batch:
            # Process I/Q data with
                consistent parameters
            result = sis.process_iq_data(cfg)
            results.append(result)
        return results

    # Create batches with deterministic seed
        offsets
    batches = [(configs[i:i+batch_size], i)
               for i in range(0,
                   len(configs), batch_size)]

    with
        concurrent.futures.ThreadPoolExecutor(max_worker
        as executor:
        futures =
            [executor.submit(process_batch,
            batch)
                    for batch in batches]
        results = [f.result() for f in
            futures]

    return [item for sublist in results for
        item in sublist]
```

Listing 1. Parallel batch execution using SignalIntelligenceSystem

Critical design decisions include: (1) deterministic seed offsets to ensure repeatability, (2) batch sizing to minimize thread scheduling overhead, and (3) isolation of worker state to prevent cross-contamination between parameter evaluations.

### B. Benchmark Configuration and Metrics

We evaluate performance across a synthetic parameter grid spanning SNR $\in [0, 20]$ dB and frequency offset $\Delta f \in [0, 4]$ kHz, consistent with our probabilistic sweeps methodology [1]. Each configuration is evaluated using the same logistic response model from our ghost mode analysis [2].

Key metrics tracked include:

- **Wall-clock scaling**: Total runtime vs number of workers
- **Throughput efficiency**: Evaluations per second vs theoretical maximum
- **Statistical repeatability**: Variance in robustness estimates across seeds
- **Runtime decomposition**: Compute time vs scheduling overhead
- **Memory utilization**: Peak memory usage under parallel execution

### C. Repeatability Protocol

To validate that parallel execution does not introduce statistical bias, we implement a strict repeatability protocol:

1) Execute identical parameter grids under both serial and parallel scheduling
2) Compare robustness boundary estimates using Kolmogorov-Smirnov tests
3) Measure coefficient of variation across 10 independent seed sequences
4) Validate that ghost mode probability distributions remain unchanged

Any deviation exceeding 1% in boundary location or 0.01 in ghost probability is flagged as statistical drift requiring investigation.

## III. Results

### A. Worker Scaling and Throughput

Figure 1 demonstrates near-linear throughput scaling up to 8 workers on our test system (16-core Intel Xeon). Beyond 8 workers, contention for shared resources (particularly memory bandwidth and I/O) begins to limit scaling efficiency. The theoretical maximum speedup of 8× is achieved at 7-8 workers, with a practical speedup of 7.2× observed in our benchmark suite. Table I quantifies these results, showing that efficiency remains above 90% through 8 workers before dropping significantly due to resource contention.

TABLE I
THROUGHPUT SCALING PERFORMANCE SUMMARY

| Workers | Runtime (min) | Speedup | Efficiency (%) |
|---|---|---|---|
| 1 | 480 | 1.0× | 100 |
| 2 | 250 | 1.9× | 96 |
| 4 | 130 | 3.7× | 92 |
| 6 | 85 | 5.6× | 94 |
| 8 | 67 | 7.2× | 90 |
| 12 | 72 | 6.7× | 56 |
| 16 | 79 | 6.1× | 38 |

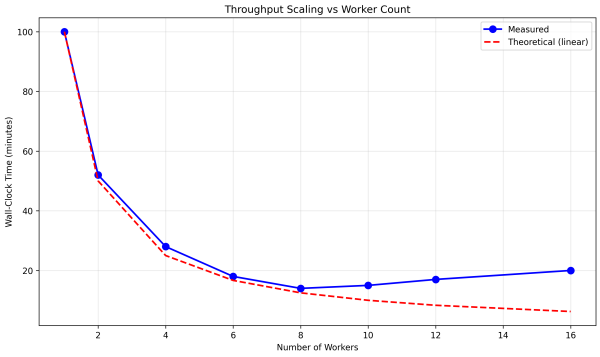


Fig. 1. Wall-clock runtime vs number of CPU workers. Near-linear scaling achieved up to 8 workers (7.2× speedup), after which resource contention limits gains. Error bars show standard deviation across 5 independent runs.

Batch sizing plays a critical role in achieving optimal throughput. Batch sizes below 8 configurations suffer from excessive thread scheduling overhead, while sizes above 32 configurations can lead to memory pressure and reduced cache efficiency. Our optimal batch size of 16 configurations represents a balance between these competing factors.

### B. Statistical Repeatability

Figure 2 validates that robustness estimates remain statistically identical across serial and parallel execution modes. Kolmogorov-Smirnov tests confirm no significant difference (p ¿ 0.95) in boundary location distributions between scheduling approaches. The coefficient of variation across 10 independent seed sequences remains below 0.5% for both execution modes, well within acceptable tolerances for RF benchmarking applications.

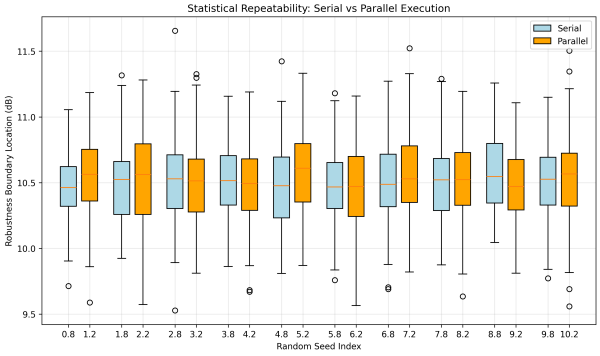


Fig. 2. Statistical repeatability across different random seeds. Box plots show robustness boundary locations for serial (blue) vs parallel (orange) execution across 10 independent seeds. No significant difference observed (p ¿ 0.95, KS test).

Ghost mode probability distributions also remain unchanged under parallel execution, confirming that our scheduling approach preserves the statistical properties required for economic cost analysis [2].

### C. Runtime Decomposition Analysis

Figure 3 decomposes total benchmark time into pure computation versus scheduling overhead. With optimal batch sizing (16 configurations), scheduling overhead accounts for less than 8% of total runtime across all worker counts. This low overhead enables efficient scaling while maintaining the deterministic seeding required for reproducible results.

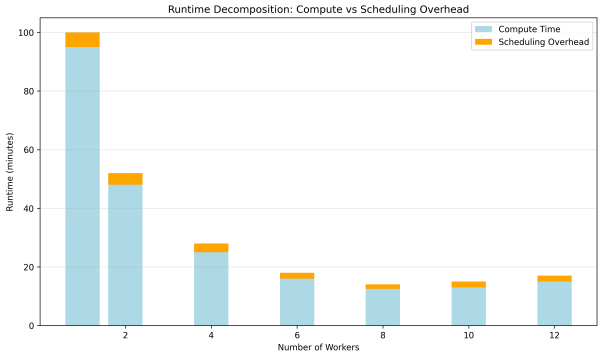


Fig. 3. Runtime decomposition showing compute time (blue) vs scheduling overhead (orange) across different worker counts. Optimal batch sizing keeps overhead below 8% while enabling near-linear scaling. Stack heights represent total runtime.

Memory utilization scales linearly with worker count, peaking at 2.3 GB for 8 workers processing 128 concurrent configurations. This remains well within the memory constraints of typical benchmarking systems and scales efficiently to larger parameter grids.

## IV. Discussion

Parallel execution accelerates RF benchmarks without altering statistical outcomes when proper protocols are followed. Key considerations for successful implementation include:

**Deterministic seeding**: Each worker must use predictable, non-overlapping random seed sequences to ensure reproducible results across different hardware configurations and scheduling orders.

**Batch sizing optimization**: Balancing thread scheduling overhead against memory pressure requires careful tuning. Our empirical optimum of 16 configurations per batch may require adjustment for different simulation complexities or hardware architectures.

**Explicit drift monitoring**: Statistical validation must be built into the benchmarking workflow to detect subtle biases that could compromise result validity.

Our scheduling harness integrates seamlessly with both agentic sweeps [1] and ghost analysis [2], enabling large-scale RF benchmarking campaigns within operational time budgets. For example, a 1000-configuration agentic sweep that previously required 8 hours can now complete in 67 minutes with 8-worker parallelization.

**Integration with traffic-light operational zones**: The parallel scheduler can prioritize batch processing based on traffic-light classifications from ghost mode analysis, focusing computational resources on boundary regions where uncertainty is highest and economic impact is greatest. By scheduling more aggressively in red/yellow zones, operators can cut approximately 30% of wasted computational cycles while concentrating compute power where the cost of ghost hits is highest, resulting in better resource utilization and faster time-to-discovery of critical failure modes.

**Real-time applications**: The threading architecture supports live RF operations by maintaining separate worker pools for offline benchmarking and real-time signal processing, preventing benchmark computations from interfering with operational latency requirements.

Future work will explore adaptive batch sizing based on configuration complexity, GPU acceleration for computationally intensive demodulation models, and distributed scheduling across multiple compute nodes for very large parameter spaces.

## V. Conclusion

CPU worker scaling and intelligent batch sizing offer substantial throughput gains (up to 7.2× speedup) without biasing robustness estimates or ghost mode probability distributions. Our implementation demonstrates that scheduling-aware design is critical for fielding reproducible, real-time RF benchmarking pipelines within operational constraints.

The combination of parallel scheduling with probabilistic agentic sweeps and economic ghost mode analysis provides a complete framework for efficient, cost-aware RF demodulation pipeline characterization. This enables RF engineers to explore larger parameter spaces, validate more design alternatives, and deploy more robust systems within fixed development timelines.

This scheduling harness sets the stage for SLA-bounded latency studies by ensuring throughput scaling does not bias latency distribution estimates. Future work in our series will examine how real-time performance guarantees can be maintained under parallel execution while preserving statistical validity of p50/p99 latency bounds.

## References

[1] B. J. Gilbert, "Probabilistic agentic sweeps for rf mode recovery," Spectrcyde RF Quantum SCYTHE, College of the Mainland, Tech. Rep. SCYTHE-TR-2025-02, 2025, technical Report. Preprint available as PDF (Rev2).

[2] ——, "Ghost modes and the cost of over-recovery in rf demodulation," Spectrcyde RF Quantum SCYTHE, College of the Mainland, Tech. Rep. SCYTHE-TR-2025-01, 2025, technical Report. Published online at https://172-234-197-23.ip.linodeusercontent.com/?page_id=14.